# Virtualization within FermiGrid

Keith Chadwick
??-Dec-2009

Abstract:

This document discusses the virtualization deployment within FermiGrid.

**Document Revision History:**

| Version | Date | Author | Comments |
|---|---|---|---|
| 0.1 | 27-Nov-09 | Keith Chadwick | Initial highly draft version. |
| 0.2 | 04-Dec-09 | Keith Chadwick | Draft version. |
| 1.0 | 14-Jan-2010 | Keith Chadwick | Version placed in CD-DocDB |
| | | | |

**Table of Contents:**

<insert future TOC here>

## Charge from Steve Wolbers

I have a request for a document that analyzes the use of VM's and the impact that the use of the VM's has had and may in the future. In particular I am looking for an analysis of the use of VM's in Fermigrid and the impact on the number of physical machines required to run the services. I don't have a specific charge or request so leave it up to you to structure the document. I'm primarily looking for information but not necessarily a detailed analysis. At some point in the future we may ask for a more thorough and wider discussion but for now I am asking just a couple of people to do some work and we'll take it from there.

## Introduction

Virtualization is a term that refers to the abstraction of computer resources. Virtualization has evolved over time and has included memory virtualization, as was implemented on the Digital VAX (Virtual Address eXtension) systems, storage virtualization (enStore), and most recently (for the purposes of this document) virtualization of the x86 platform. There are two key components to the x86 virtualization:

1. Host operating system (also known as “Dom-0” or “hypervisor”).

2. Guest operating system (also known as “Dom-U”).

There are several configuration options in both the hardware and host operating system that can affect the performance of virtualization:

1. Full Virtualization - is a virtualization technique used to provide a virtual machine environment that is a complete simulation of the underlying hardware. In full virtualization, any software capable of execution on the raw hardware can be run in the virtual machine and, in particular, any operating system. Full virtualization would allow a Linux system with Xen hypervisor to run a Microsoft Windows guest operating system (or vice-versa).

2. Paravirtualization - is a virtualization technique used to provide a virtual machine environment that is a partial simulation of the underlying hardware. Most of the CPU instructions are executed on the directly on the underlying hardware, those instructions that can change the machine “state” (context switching) are intercepted and emulated. The result is that the performance of the guest operating system can rival (or in some specialized cases exceed) the performance of the operating system executing on the “bare metal”. This improved performance under paravirtualization does require that the guest operating system be “virtualization aware”. In the case of Linux, specialized Kernels must be loaded in both the Dom-0 and Dom-U systems. Paravirtualization also precludes running significantly different operating system flavors (Linux vs. Microsoft Windows) unless the guest operating system can be patched to become “paravirtualization” aware.

3. Hardware Assisted Virtualization - is a platform virtualization approach that enables efficient virtualization using help from hardware capabilities. Both Intel (Intel VT) and AMD (AMD-V) added specialized instructions in their CPUs to assist with x86 hardware virtualization back in 2005-2006. Both Full and Para virtualization can make use of capabilities provided by hardware-assisted virtualization (when present).

4. Operating system (chroot) virtualization - is a virtualization method where the kernel of an operating system allows for multiple isolated user-space instances, instead of just one. On Unix systems, this technology can be thought of as an advanced implementation of the standard chroot mechanism. In addition to isolation mechanisms, the kernel often provides resource management features to limit the impact of one container's activities on the other containers.

In the area of the operating system hypervisor there are several options currently in the market place. Among them are:

| **Virtualization Product** | **Product Licensing** | **Virtualization "Vendor" Product URL** |
|---|---|---|
| VMware | Commercial | www.vmware.com |
| Xen | Open Source | www.xensource.com |
| KVM | Open Source | http://www.linux-kvm.org/page/Main_Page |
| Hyper-V | Commercial | http://www.microsoft.com/hyper-v-server/en/us/default.aspx |
| Oracle VM | Commercial | http://www.oracle.com/us/technologies/virtualization/index.htm |

The virtualization technology that is used within FermiGrid is currently Xen. This was selected because it is "Open Source" and that the upstream vendor (Redhat) has built in support for Xen virtualization in the Linux operating system.

## Background

The FermiGrid project began in late 2004 with the charge from the Computing Division Head to deploy a centrally support set of common Grid services (initially VOMRS, VOMS and GUMS) and to encourage the migration from siloed (experiment specific) offline analysis clusters to a Grid based infrastructure that would allow sharing of resources.

Based in the initial scope, three (3) systems were acquired. The systems that supported the VOMS and GUMS services were commissioned and formally released to operation on 01-Apr-2005.

| System | System Specification | Role | Commission Date |
|---|---|---|---|
| fermigrid1 | Dell 2850, 3.6 GHz | Site Gateway | |
| fermigrid2 | Dell 2850, 3.6 GHz | Server to host the Virtual Organization Member Registration Service (VOMRS) & Virtual Organization Management Service (VOMS) | 01-Apr-2005 |
| fermigrid3 | Dell 2850, 3.6 GHz | Server to host the Grid User Mapping Service (GUMS) | 01-Apr-2005 |

Later in 2005 and early 2006, two (2) additional systems were acquired:

| System | System Specification | Role | Order Date | Delivery Date | Commission Date |
|---|---|---|---|---|---|
| fermigrid4 | Dell 2850, 3.6 GHz | Server to host the Site AuthoriZation (SAZ) service | 28-Jun-2005 | 20-Jul-2005 | 01-Oct-2006 |
| fermigrid0 | Dell 2850, 3.6 GHz | Server to host FermiGrid Monitor and Metrics collection / hot spare | 06-Jan-2006 | 18-Jan-2006 | 16-Feb-2006 |

Later in 2006 two additional services were deployed on fermigrid4:

1. MyProxy (to allow the first implementation of the Fermilab site gateway).

2. Squid (to support the caching of certificate revocation lists).

Based on the operational experience with the above set of Grid services, coupled with the desires of the user community, and the recognition that an outage of either the single GUMS or SAZ server had the potential to "turn the Grid at Fermilab off", the FermiGrid High Availability (FermiGrid-HA) project was started in <insert date> to eliminate as many single point of failure as possible for the core FermiGrid services (VOMS, GUMS and SAZ). In the initial planning of the FermiGrid-HA project, failures of the building (power & environment) or network were defined as "out of scope" for the FermiGrid-HA project; addressing these types of failures would be addressed by a future FermiGrid Resilient Services (FermiGrid-RS) project. Even though building and network failures were specified as "out of scope" for FermiGrid-HA project, when the potential designs and implementations for the deployment of the FermiGrid-HA services were discussed, careful consideration was made regarding the impact of the design and/or implementation on the future FermiGrid-RS project.

Two options were identified for the deployment of the FermiGrid-HA services:

| Deployment Option | Deployment Description |
|---|---|
| Active-Standby | The active copy of the service handles all requests, the standby copy or copies of the service do not handle any requests. |
| Active-Active | All copies of the service are continuously handling requests. |

The decision was to deploy the FermiGrid-HA services in an Active-Active configuration to the maximum extent possible for the following reasons:

1. Reduction of single points of failure.

2. All copies of a particular service would be available to respond to the incoming service requests under normal operation.

3. Individual service function can be continuously and independently verified.

4. In the event of an individual service failure the remaining copy would already be properly configured and in operation to pick up the load of the incoming service requests.

The initial design architecture for FermiGrid-HA was:

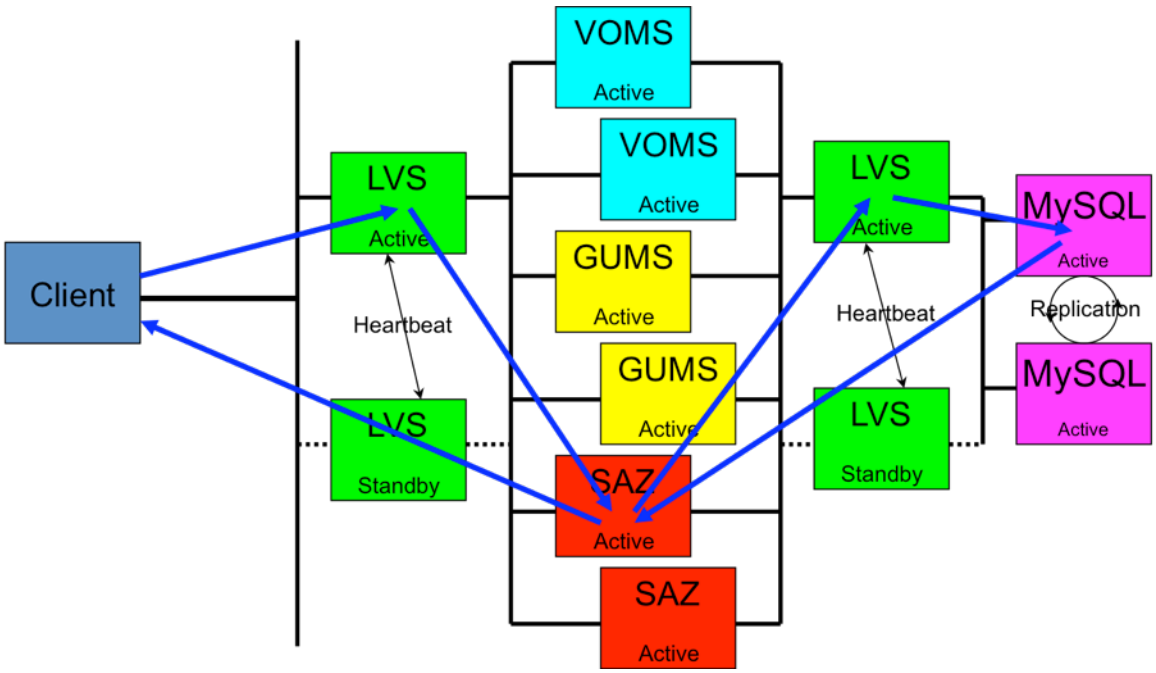
1. A Linux Virtual Server (LVS) frontend deployed in an Active-Standby configuration.

2. Individual Grid services deployed in an Active-Active configuration on a minimum of two independent systems.

3. All Grid services configured to use a common MySQL backend cluster.

4. The MySQL backend cluster deployed using an Active-Active configuration on two independent systems using MySQL V5 with multi master database replication.

Based on this architecture, the minimum number of systems to deploy FermiGrid-HA would be a total of ten (10) systems - two (2) systems each for LVS, VOMS, GUMS, SAZ and MySQL. This is shown in the service functional communication diagram below:




The available budget (approximately $20K) would not support the acquisition of the number of physical systems that would be required to implement this architecture. Fortunately, FermiGrid personnel had been researching and experimenting with Xen based Virtualization under Scientific Linux Fermi (SLF) V4, driven by to the potential future need to offer support for Grid "edge services" by various high profile VO's (CMS and Atlas).

It was realized that if Xen based Virtualization was incorporated into the FermiGrid-HA architecture; the result would be that only two systems would need to be acquired to deploy the FermiGrid-HA services. With the incorporation of Xen based Virtualization, the revised design architecture for FermiGrid-HA became:

1. Scientific Linux Fermi (SLF) V5.0 with Xen Virtualization with a minimum of ten (10) virtual machines spread across two (2) physical systems.

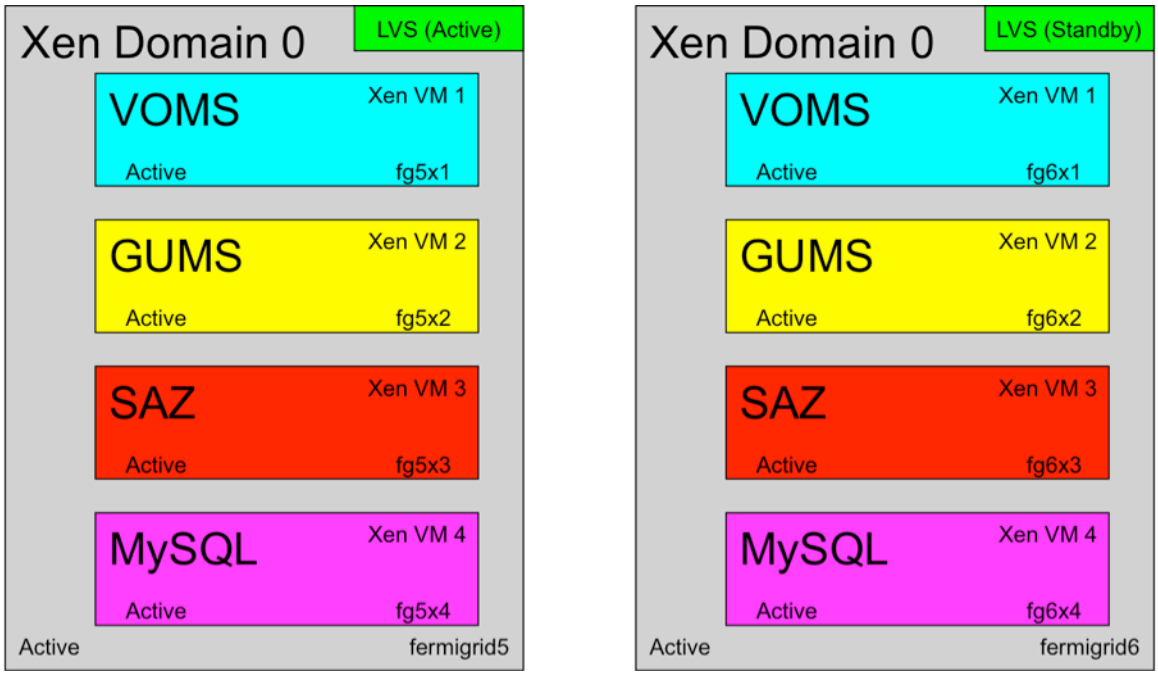
2. A Linux Virtual Server (LVS) frontend deployed in an Active-Standby configuration.

3. Individual Grid services would be deployed in an Active-Active configuration on a minimum of two independent systems.

4. All Grid services would be configured to use a common MySQL backend cluster.

5. The MySQL backend cluster would be deployed using an Active-Active configuration on two independent systems using MySQL V5 with multi master database replication.

This architecture would require a minimum of two (2) physical systems to implement and would also support (with some extensions to the LVS and MySQL configurations) the future FermiGrid-RS project. The system requirements were in good agreement with the available budget, so the decision was made to proceed to the acquisition, commissioning and deployment of this architecture. The expected virtual machine "layout" was shown the figure below:




## FermiGrid-HA Commissioning

The two systems necessary to deploy FermiGrid-HA were ordered on 25-Apr-2007, delivered on 18-May-2007, and installed in early Jun-2007, and then underwent the following steps to complete the deployment:

1. Burn-In of fermigrid[5,6] hardware.
2. Installation and configuration of SLF V5.0 on fermigrid[5,6].
3. Installation and configuration of Xen hypervisor together with the necessary virtual machines.
4. Installation and configuration of MySQL V5.
5. Installation and configuration of the VOMS, GUMS and SAZ Grid services.
6. Installation and configuration of LVS.

Previously, Redhat had announced that Xen would be included in Redhat V5.0. Since SLF is built from the Redhat source, it was expected that SLF V5.0 would be a good operating system platform on which to base FermiGrid-HA. Unfortunately, following the installation of SLF 5.0 onto the fermigrid[5,6] hardware, it was discovered that the Xen components that were included in SLF 5.0 did not properly function – the hypervisor was unstable and the utilities to create and manage the virtual machines did not function. To address this, Steve Timm downloaded the current Xen virtualization source from XenSource and, with the help of the community support from the **xen-users** discussion list, was able to build a custom SLF 5.0 kernel with the necessary Xen modifications and the minimum set of Xen VM management utilities.

Once the hardware together with the physical and virtual systems was successfully deployed, Dan Yocum was able to install MySQL, the Grid services and LVS. This work faced another set of challenges, first to demonstrate the MySQL V5 multi-master (circular) replication, second to install the Grid services and configure them to use the MySQL backend, and finally to configure LVS to manage the access to the Grid services.

All of the above steps were completed by the beginning of Oct-2007.

As part of the FermiGrid-HA project acceptance testing, a suite of functionality and performance tests was performed against the FermiGrid-HA deployment in Oct-2007. These tests included basic service performance, service failover testing as well as service stress (capacity) testing:

| **FermiGrid-HA Test** | **FermiGrid-HA Test Description** |
|---|---|
| Basic service performance | Verification that the service is performing as expected. In the case of active-active services, verify that all copies of the service are responding to service requests. |
| Service failover | Verify that the service fails over. Verify that all service requests are being routed to the surviving service instances. |
| Service stress (capacity) test | Run sufficient copies of the service client to determine the total capacity of the service under normal operations (service calls/day). |

These tests were performed on the "final" configuration. The results of the tests were extremely positive – the LVS and the MySQL-HA services performed as expected, the GUMS service demonstrated ~10,000,000 calls/day and the SAZ service demonstrated ~1,000,000 calls/day.

The load on the GUMS service VMs during the stress test was ~9.5 and the CPU idle time was 15%. The load on the LVS and backend MySQL-HA service VMs during the stress test was very light (load under 1 and the CPU idle time was 92%). GUMS uses the java hibernate method which caches database records – that is why the backend MySQL database VMs load is so light compared to the GUMS service VMs. Based on these

measurements, we'll need to start the planning for a third GUMS server in FermiGrid-HA when we hit the ~7.5M mappings/day mark.

The load on the SAZ service VMs during the stress test was ~12 and the CPU idle time was 0%. Again, the load on the LVS and backend MySQL-HA service VMs during the stress test was very light (load under 2 and the CPU idle time was 98%). Based on these measurements, the expectation was that we would need to start the planning for a third SAZ server in FermiGrid-HA when we hit the ~0.75M mappings/day mark.

FermiGrid-HA formally commenced operations on 01-Dec-2007.

Following the successful deployment of FermiGrid-HA for the core FermiGrid services, an analysis was performed on the other services operated by FermiGrid. Based on this analysis a decision was made to extend the FermiGrid-HA project in order to deploy all services operated by FermiGrid into virtual machines, with various strategies to deploy "High Availability" (HA) services:

1. Trivial monitoring or information services (such as Ganglia, Nagios and Zabbix) are deployed on two independent virtual machines.

2. Services that natively support HA operation (Condor Information Gatherer, FermiGrid internal ReSS deployment) are deployed in the standard service HA configuration on two independent virtual machines.

3. Services that maintain intermediate routing information (Linux Virtual Server) are deployed in an active/passive configuration on two independent virtual machines. A periodic heartbeat process is used to perform any necessary service failover.

4. Services that do not maintain intermediate context (i.e. are pure request/response services such as GUMS and SAZ) are deployed using a Linux Virtual Server (LVS) front end to active/active servers on two independent virtual machines.

5. Services that support active-active database functions (circularly replicating MySQL servers) are deployed on two independent virtual machines.

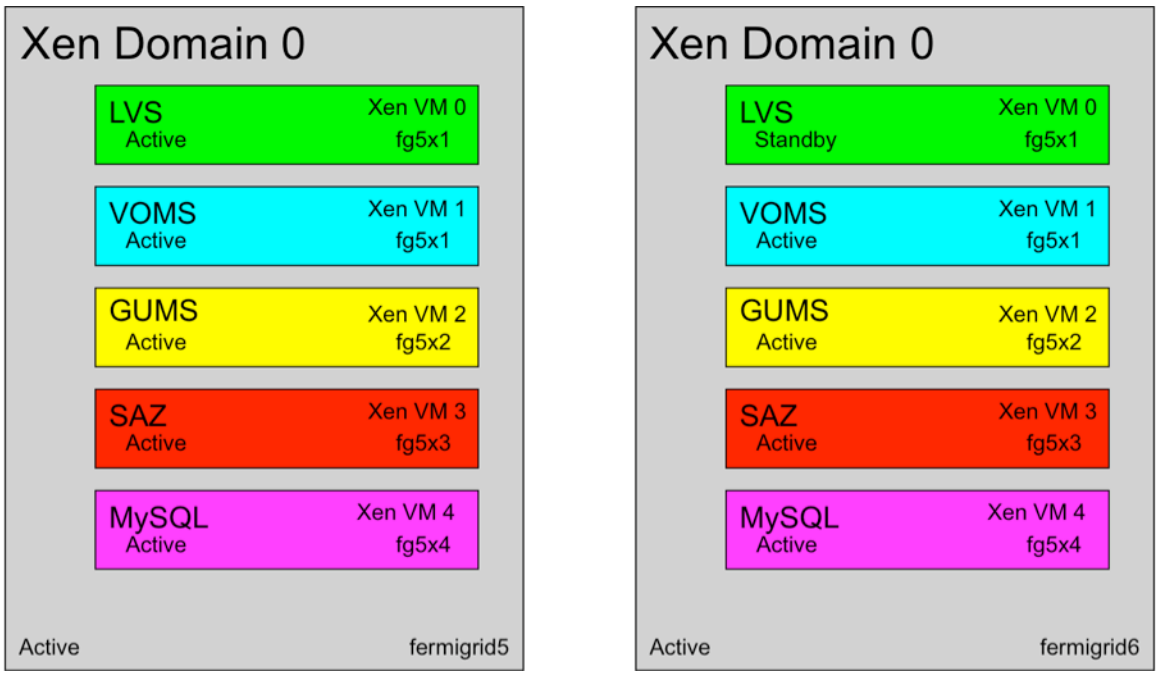
An additional goal of the FermiGrid-HA extension was to deploy the Globus Gatekeeper and MyProxy services as HA services. This goal has proven significantly harder to implement than initially anticipated. With the recent hiring of Faarooq Lowe, progress is finally being made – an initial MyProxy-HA deployment has been successfully demonstrated on two test systems using the Distributed Replicated Block Device (DRBD) driver from drbd.org (http://www.drbd.org/) and work is currently underway to write up a formal deployment proposal for MyProxy-HA onto the production service infrastructure.

Once MyProxy-HA has been commissioned on the production service infrastructure, the focus will turn to the development of Gatekeeper-HA functionality for the FermiGrid Site Gateway (fermigridosg1).

## Operational Experience

The operational experience with the initial FermiGrid-HA uncovered that the IP V6 drivers that were included Xen virtualization was vulnerable to IP V6 packets. The solution was simple and straightforward – disable the IP V6 drivers in the Xen Kernels on the FermiGrid systems.

LVS was initially deployed on the base hypervisor "Dom-0" systems. Operational experience showed that it could become resource starved when running on the Dom-0 and the recommended solution was to move LVS out of the Dom-0 into a pair of dedicated Dom-U's. This configuration change was deployed on 06-Mar-2008.




In June 2008, the hardware that was running the fcdfosg1 gatekeeper became unstable and was frequently crashing. As this server was well past the warranty period and was already slated for replacement, a decision was undertaken to attempt to convert the standalone system into a Xen dom-U. In the space of ~4 hours, Steve Timm was able to archive the operating system files and re-install them as a paravirtualized guest operating system to restore the gatekeeper functionality.

Operational experience with the SAZ service has shown that it is vulnerable to a denial of service attack as a consequence of an "authorization tsunami". Investigation has shown that the actual limitation is in the SAZ server code management of the database

connections and the management of the number of threads to process client requests. The limitation is not in the SAZ client, nor is it in the backend MySQL database. To mitigate the impact of the SAZ server response to "authorization tsunamis", additional SAZ servers have been deployed to provide isolation of major client communities from "authorization tsunamis" in other client communities. At the present time, each major customer has a dedicated pair of (virtualized) SAZ-HA servers to respond to SAZ service requests:

| Client Community | SAZ-HA Servers |
|---|---|
| Grid Cluster gatekeepers | fg[5,6]x3 |
| CDF Grid Cluster 1,2,3 Worker Nodes | fcdf[1,2]x4 |
| CDF Sleeper Pool | fcdf[3,4]x3 |
| D0 Grid Cluster 1,2 Worker Nodes | d0osg[1,2]x4 |
| US CMS T1 Cluster Worker Nodes | fg[5,6]x5 |
| GP Grid Cluster Worker Nodes | fg[1,4]x5 |

The rapid deployment of the partitioned SAZ-HA servers was only possible due to the prior investment of Xen based virtualization by FermiGrid personnel.

## Other Operational Experience

While not part of the formal FermiGrid-HA architecture, FermiGrid has also successfully demonstrated operation of multiple versions and configurations of operating systems on a single common virtual machine hypervisor:

| Role | Operating System |
|---|---|
| Hypervisor | 64bit SLF 5.3 Xen Dom-0 |
| Virtual Machine 1 | 32bit SLF 4.7 |
| Virtual Machine 2 | 64bit SLF 4.7 |
| Virtual Machine 3 | 32bit SLF 5.3 |
| Virtual Machine 4 | 64bit SLF 5.3 |

While it is possible to operate this configuration with minimal issues, it should be noted that FermiGrid configures all of the virtual deployments to use paravirtualization for optimum performance. As a consequence of this paravirtualization configuration, the SLF 4 virtual machines that would normally use Kernel versions from 2.6.9 series wind up using the Kernel versions from 2.6.18 series. This caused some minor issues with the nash shell that is used at system start – processes that use this shell occasionally fail to terminate as expected and "runaway" which impacts performance – the mitigation for this is that FermiGrid runs an automated process to terminate and clean up after any "runaway" copies of the nash shell once the SLF 4 systems have completed startup.

It should be noted that if FermiGrid had chosen to use full virtualization (with the attendant performance penalty), then this would not be an issue.

# Current Deployment

At the present time (01-Dec-2009), FermiGrid operates the following systems:

| System | Virtualization Technology | # of Physical Systems | # of Virtual Machines | # of Services |
|---|---|---|---|---|
| fermigrid[0-6] | Xen | 7 | 8+(6*6) = 44 | |
| gratia[1-3,5-9] | Xen | 8 | 2+3+5+(2*6) = 22 | 2*(4+1) = 10 |
| gratia[10-13] | Xen | 4 | 4*6 = 24 | 2*(4+1) = 10 |
| ress[1,2] | Xen | 2 | 2*5 = 10 | 2 |
| fcdf[0-5] | Xen | 6 | 4+5+5+4+4+4 = 26 | 5+3 = 8 |
| fcdft[0-2] | Xen | 3 | 4+4+5 = 13 | 2+1 = 3 |
| d0osgsrv[1,2] | Xen | 2 | 5+5 = 10 | 2 |
| gp (fnpcsrv[3-5]) | Xen | 3 | 3*4 = 12 | 3+1 = 4 |
| fgtest[0-1,3-5] | Xen | 5 | 8+7+7+7+7 = 36 | <varies> |
| fgitbse | Xen | 1 | 1 | 1 |
| gw??? | kvm | 5 | 5 | 1 |
| | | | | |
| **Total:** | | **46** | **203** | |

| Service | # Servers | Virtual Server Names | Physical Host Names |
|---|---|---|---|
| LVS | 2 | fg[5,6]x0 | fermigrid[5,6] |
| VOMS | 2 | fg[5,6]x1 | |
| GUMS | 2 | fg[5,6]x2 | |
| SAZ (Gatekeepers) | 2 | fg[5,6]x3 | |
| MySQL | 2 | fg[5,6]x4 | |
| SAZ (CMS WN) | 2 | fg[5,6]x5 | |
| | | | |
| <unused>/MyProxy | 1+1 | fg[1,4]x0 | fermigrid[1,4] |
| Site gateway | 1+1 | fg[1,4]x1 | |
| Info Gatherer | 2 | fg[1,4]x2 | |
| Condor Master | 2 | fg[1,4]x3 | |
| OSG-TG Gateway | 1+1 | fg[1,4]x4 | |
| SAZ (GP WN) | 2 | fg[1,4]x5 | |
| | | | |
| <unused> | 2 | fg[2,3]x0 | fermigrid[2,3] |
| syslog-ng | 2 | fg[2,3]x1 | |
| Ganglia/zabbix/nagios | 2 | fg[2,3]x2 | |
| Squid | 2 | fg[2,3]x3 | |
| Future MyProxy | 2 | fg[2,3]x4 | |
| <unused> | 2 | fg[2,3]x5 | |

## Security Considerations

Secure operation of the FermiGrid resources requires that the FermiGrid Services and Operations team members must be diligent in their management of the (physical and virtual) systems that host or provide the Grid services.

The principal mechanism to accomplish this is through the nightly yum updates and (approximately) monthly Kernel updates from the Scientific Linux Fermi repositories. However it is recognized that Virtualization = More "moving parts" = More opportunity for an attacker to gain access through a compromise (such as an attack on the virtualization software).

There are specific attacks that target virtualization. An example of such an attack is the "Blue Pill" exploit (see http://blackhat.com/presentations/bh-usa-06/BH-US-06-Rutkowska.pdf) that relies on the Secure Virtual Machine (SVM) instruction extensions present in the Advanced Micro Devices Inc. (AMD) 64-bit AMD Athlon processors. With SVM, software developers are able to manipulate processor registers, interrupts, input/output and so on for virtual machine functionality at the hardware level. The "Blue Pill" attack manipulates kernel mode memory paging and the VMRUN and related SVM instructions that control the interaction between the host (hypervisor) and guest (virtual machine). This permits undetected, on-the-fly placement of the host operating system in its own secure virtual machine allowing for complete control of the system including manipulation by other malware. While initially targeted against AMD processors, the "Blue Pill" exploit has been extended to also target the Intel VT series of processors.

The first line of defense is assuring that the operating systems are well maintained – the OS patches and utilities are up to date for both the host and guest operating systems.

If the host and guest operating systems are well maintained, it is possible to consider running older copies of the guest operating system that are compatible with the sharing of the Kernel (if configured for paravirtualization). As indicated above, FermiGrid has demonstrated successful and secure operation of guest operating systems running 32 and 64 bit SLF4 under a host operating system running 64 bit SLF5.

In the case where support for the guest operating system is no longer available (as will eventually be the case with SLF4), it is possible to utilize virtualization to continue to offer a subset of the functionality, that would be available if the support for the guest operating system were still available, using the following mitigation strategy:

1. The services offered by the obsolete guest operating system should be reduced to the absolute minimum set of services required for the role of the guest operating system.

2. Direct "shell" (interactive) access to the obsolete guest operating system from the general Internet shall not be available.

3. "Shell" (interactive) access on the obsolete guest operating system shall only be available through the following procedure – The user initializes their regular credentials (a bare "kinit"), logs into the "shell" on a fully patched "bastion" system (such as the host system), initializing a set of special credentials on the "bastion" system (example: "kinit <user>-<hostname>"), and then accessing the obsolete guest operating system using these special credentials.

The above mitigations will allow rapid detection of stolen credentials:

1. If an attempt is made to log into the "obsolete" system using a copy of the initial set of user credentials that were used to log into the "bastion" system, then this action will trigger a security alert in both the "obsolete" system logs and the Kerberos domain controller logs.

2. If an attempt is made to log into the "bastion" system using a copy of the special credentials for the "obsolete" system, then this action will trigger a security alert in both the "bastion" system logs and the Kerberos domain controller logs.

3. In addition, the set of remote systems that are logging into the "bastion" system should be tightly controlled and monitored.

Both the Kerberos domain controller logs, and the operating system logs (provided the operating systems are configured to send copies of their logs to the CST central syslog service), can be automatically scanned for this signature, and appropriate alarms triggered if such a compromise attempt is found.

# Future

Redhat has announced that kvm based virtualization will eventually be their "chosen" flavor of virtualization; FermiGrid has an active investigational activity to explore kvm virtualization to gain experience with the operation of kvm based virtualization.

Under Redhat V5.3, kvm is identified as a "technology preview", and the operational experience of FermiGrid with the virtualization available in Redhat V5.3 is that the performance and stability of kvm based virtualization is somewhat less that of Xen based virtualization. This is similar to the prior experience with Xen based virtualization - while Redhat V5.0 claimed to be able to support Xen virtualization, actual functional Xen virtualization was only delivered "out of the box" in Redhat V5.2.

Nevertheless, as part of the FermiGrid virtualization investigation activity, we have identified the necessary configuration changes to switch from Xen based virtualization to kvm based virtualization. Based on this research, we believe that it should be relatively straightforward to convert the current production Xen based virtualization deployment to a kvm based virtualization deployment via an automated script.

## Conclusions

FermiGrid has made extensive use of Xen Virtualization both in direct support of High Availability services (FermiGrid-HA) and to allow server consolidation. The deployment of Xen Virtualization has allowed FermiGrid to implement ~4-8x server consolidation. This rate of server consolidation is dominated by the FermiGrid-HA requirement to offer at least two independent copies of any service. If the requirement to offer two independent copies of the services was dropped, then the FermiGrid virtualization infrastructure would be delivering ~6-16x server consolidation.

FermiGrid has also been able to deploy multiple operating systems and configurations on a single physical system (albeit using paravirtualization under Xen).

FermiGrid will continue to monitor the maturity and developments in the Redhat distribution to determine the proper time to schedule the transition from Xen based virtualization to kvm based virtualization.